Labor für Künstliche Intelligenz
Forschung, Entwicklung, Beratung, Ausbildung

CTL
A description logic with expressive concrete domains

Gerd Kamp, Holger Wache

May 1996

LKI-M-96/01

Labor für Künstliche Intelligenz
Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Str.30, 22527 Hamburg, Germany

CTL
A description logic with expressive concrete domains

Gerd Kamp, Holger Wache

LKI-M-96/01

This research has partially been supported by the DFG under the grant DFG Ne/279

CTL
A description logic with expressive concrete domains

Gerd Kamp
University of Hamburg
Vogt-Kölln-Str.30
22527 Hamburg
kamp@informatik.uni-hamburg.de

Holger Wache
DFKI GmbH
P.O. Box 2080
67608 Kaiserslautern
wache@dfki.uni-kl.de

Abstract

Compared with frame-based systems, description logics have the advantage of well-defined semantics and powerful inferences. In order to exploit these advantages in technical domains, the ability to use concrete domains is needed, e.g. systems of (in)equalities over (non)linear polynomials to handle physical laws. Existing systems can only cope with comparisons between attributes.

We present an approach that considerably improves the expressiveness of the concrete domains. CTL [1] is based on the ideas presented in [1] and [5]. Concrete domains are realised through a well-defined interface to external algorithms. Constraint Logic Programming (CLP) systems allow us to easily realise a whole range of concrete domains, e.g. over sets of symbols and numbers. In particular, we are able to handle systems of arbitrary linear polynomials. They also enable us to automatically participate in recent and future improvements in the areas of CLP and computer algebra, e.g. systems capable of handling arbitrary non-linear polynomials.

1 Introduction

In order to use description logics (DL) in technical domains expressive description languages are needed. In addition to the abstract domain, several concrete domains such as numbers, strings and symbols must be added to the language and the inferences of the description logic.

Current representation systems such as LOOM or CLASSIC take this into account by incorporating access to host data types. They realise mainly a single concrete domain over

[1]Configurable (or Constraint-based) Terminological Logic

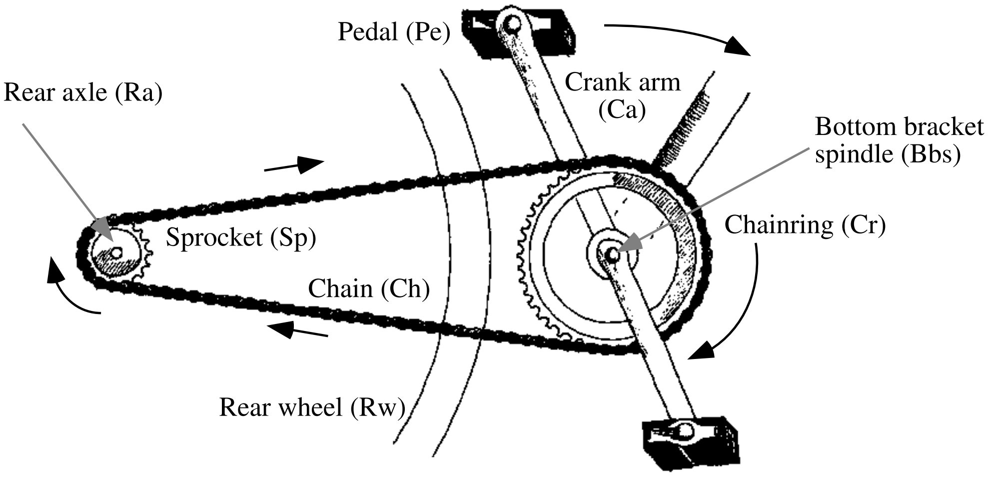

Figure 1: A simple bike drivetrain

numerical values. But only comparisons between numerical attributes and numerical constants are taken into account when drawing the TBox inferences, and these inferences are themselves incomplete. In contrast to that, Kris and Taxon have sound and complete algorithms. Both use a scheme for the integration of concrete domains developed by Baader and Hanschke [1, 5]. But they also realise only a single concrete domain allowing only comparisons between numeric attributes. Hence this is only a slight improvement with respect to Loom and Classic . To summarise, there is a need for improvement along two dimensions:

1. Addition of new concrete domains, such as strings and symbols.
2. Enhancement of the expressiveness of an existing domain (i.e. to date only the numbers) by introducing new predicates.

In this paper we present Ctl , a scheme for improving in both dimensions.

2 Example

In order to apply description logics to technical applications, one must at least be able to represent the subject of the application, i.e the technical device or assembly. In this section we give a simple, but typical example of such an assembly. The task is to model the simple bike drivetrain shown in Figure 1. Our focus is on the representation of the underlying physical (mechanical) laws. The bike transmission as a special mechanism can be modeled as a set of connected links. The behaviour of mechanisms is largely determined by the relations between the intervening forces and torques. These relations are given by physical laws that apply to all mechanisms, making the drivetrain a good example for mechanical devices in general. For our purposes it is sufficient to treat the forces and torques as scalar entities. We also use numerical equations instead of quantity equations to describe the physical laws[2]. In order to describe the drivetrain, the representation of the following physical laws is necessary:

- The torques of two links rotating on the same shaft are equal.

$$M_{Rot_1} = M_{Rot_2} \tag{1}$$

- A tension link (e.g. a chain) transmits forces along the direction of the tension, i.e. the force applied to two gear wheels driven by the same chain is equal.

$$F_{Tension} = F_{Rot} \tag{2}$$

- The torque of a rotational link is the product of the applied force and the radius of the link.

$$M_{Rot} = F_{Rot} \cdot r_{Rot} \tag{3}$$

- The force applied to a link is positive.

$$F_{Link} \geq 0 \tag{4}$$

- the radius of a rotational link strictly positive.

$$r_{Link} > 0 \tag{5}$$

It is important to notice that the right-hand side of Equation 3 is a quadratic polynomial. It becomes linear when either F or r is bound. This is usually the case during ABox reasoning but not during TBox reasoning processes. The drivetrain depicted in Figure 1 can therefore be described with the following system of nonlinear inequalities[3]:

$$\begin{array}{ccc} M_{Ca} = F_{Ca} \cdot r_{Ca} & F_{Sp} = F_{Ch} & \\ M_{Ca} = M_{Bbs} & M_{Sp} = F_{Sp} \cdot r_{Sp} & F_{Ca}, F_{Cr}, F_{Bbs}, F_{Ch}, \\ M_{Bbs} = F_{Bbs} \cdot r_{Bbs} & M_{Ra} = M_{Sp} & F_{Sp}, F_{Ra}, F_{Rw} \geq 0 \\ M_{Cr} = M_{Bbs} & M_{Ra} = F_{Ra} \cdot r_{Ra} & r_{Ca}, r_{Cr}, r_{Bbs}, \\ M_{Cr} = F_{Cr} \cdot r_{Cr} & M_{Rw} = M_{Ra} & r_{Sp}, r_{Ra}, r_{Rw} > 0 \\ F_{Ch} = F_{Cr} & M_{Rw} = F_{Rw} \cdot r_{Rw} & \end{array} \tag{6}$$

In general, technical applications impose the following requirement[4]:

> "*The T- and ABox reasoning services of description logics for technical applications must take (non)linear systems of (in)equalities into account.*"

[2]This is possible if every quantity is denoted using its base unit.
[3]Neglecting the pedal.
[4]The representation of the geometry or other aspects leads to the same requirement.

3 Admissible concrete domains

Is it possible to develop such systems without losing decidability of the description language? To answer this question we first give a formal definition of (non)linear systems of inequalities. We then briefly introduce the concept of "admissible concrete domains" as developed by Baader and Hanschke [1, 5] and show that systems of (non)linear (in)equations are an admissible concrete domain by reducing the problem to the decidability of the theory over the elementary algebra of the reals [15].

Definition 3.1
A *univariate Polynomial* over a ring R with 1 (a field) is an expression

$$p(x) = \sum_{0 \leq i \leq n} a_i x^i.$$

The *degree* of the polynomial is defined by

$$\deg(p) := max\{\nu \in \mathbb{N} : a_\nu \neq 0\}.$$

Let $i = (i_1 \ \ldots \ i_n), n \in \mathbb{N}$ be a multiindex, and $|i| = i_1 + \cdots + i_n$ be the *order* of i. A *multivariate Polynomial* over R in $x = (x_1, \ldots, x_n)$ of *degree* m is defined by

$$p(x) = p(x_1, \ldots, x_n) = \sum_{|i| \leq m} a_{i_1 \ \ldots \ i_n} x_1^{i_1} \cdot \cdots \cdot x_n^{i_n}.$$

Remark 3.1

(i) Polynomials with a degree > 1 are generally called *non-linear*, Polynomials of degree $0, 1, 2, 3$ respectively *constant, linear, quadratic* or *cubic*.

(ii) In our case we are interested in polynomials with rational coefficients, i.e. in $\mathbb{Q}[x_1, \ldots, x_n]$.

(iii) The a_i are called *coefficients*, the x_i *indeterminates*.

Definition 3.2
A *System $\mathcal{U}$ of inequalities of degree n in the variables x* is a finite set of equalities and inequalities in x between polynomials of degree at most n. I.e. $U = \{g_1, \ldots, g_m\}$, with $g_i = p_i \circ q_i, \circ \in \{<, >, =, \neq, \leq, \geq\}$ and $\deg(r) \leq n, r \in \{p_1, \ldots, p_m\} \cup \{q_1, \ldots, q_m\}$.

As we have seen in Section 2, technical applications need concrete domains. In contrast to most existing description logics, Baader and Hanschke developed a scheme for integrating concrete domains into description languages rather than describing a particular extension by some specific concrete domain. Additionally this scheme has the following properties:

1. The formal semantics as well as the combination of the algorithms are given on the scheme level.

2. The resulting algorithms for the extension should not only be sound but also complete.

Definition 3.3
A *concrete domain* $\mathcal{D}$ is a relational structure, consisting of a set $dom(\mathcal{D})$, the domain of $\mathcal{D}$, and a set $pred(\mathcal{D})$, the predicate names of $\mathcal{D}$. Each $p \in pred(\mathcal{D})$ is associated with an arity n_p and an n_p-ary predicate $P^{\mathcal{D}} \subseteq dom(\mathcal{D})^{n_p}$.

Property 2 from above imposes additional constraints on the concrete domain, it must be *admissible* in order to be useful.

Definition 3.4
A concrete domain is *admissible* iff

(i) it is closed under negation (i.e. $\forall P \exists Q (Q = dom(\mathcal{D})^{n_p} \setminus P, pred(\mathcal{D})))$,

(ii) it contains a name $\top_{\mathcal{D}}$ for $dom(\mathcal{D})$ (i.e. $\forall d \in dom(\mathcal{D})(\top_{\mathcal{D}}(d) = true))$, and

(iii) the satisfiability problem of finite conjunctions is decidable. (The finite conjunction $K = \bigwedge_{i=1}^{k} p_i(\vec{x}_i)$ is *satisfiable*, iff there exists an assignment of the variables such that K becomes true).

Theorem 3.1
Let $\mathcal{D}$ be an admissible concrete domain. Then there exists a sound and complete algorithm which is able to decide the consistency problem of an ABox for $\mathcal{ALCF}(\mathcal{D})$[5].

Due to the decidability of the theory of the elementary algebra of the reals[15] it is also admissible [1].

Definition 3.5
The *theory of the elementary algebra over the reals* is the set of Terms $\mathcal{T}$ and Formula $\mathcal{F}$ over the alphabet consisting of the individual variables $V = \{x_i \in \mathbb{N}\}$ (interpreted over $\mathbb{R}$), the individual constants $0, 1$,the functional symbols $+, \cdot$, the predicate symbol $>$, the logical junctors $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$, the logical quantifiers $\exists, \forall$ and the auxiliary symbols $(,)$[6]. The Terms $\mathcal{T}$ and formulas $\mathcal{F}$ are recursively defined as usual.

Theorem 3.2
The theory of the elementary algebra over the reals is decidable.

Theorem 3.3
The theory of the elementary algebra over the reals is an admissible concrete domain.

Together with the following observations the admissibility of systems of multivariate polynomials can be concluded.

[5] TBox subsumption and a lot of other reasoning services can be reduced to this problem, see [1] for details. $\mathcal{ALC}$ is a rather expressive concept language introduced in [14]. Since TAXON as well as CTL implement this language we refer to section 4 for details.

[6] In current systems one would use a more extensive alphabet providing additional predicate symbols and all $q \in \mathbb{Q}$ as individual constants. Tarski used the alphabet $\{1, 0, -1, x_i, y_i, z_i, +, \cdot, =, >, \neg, \vee, \wedge, \exists\}$.

Remark 3.2

(i) Multivariate Polynomials $p(\vec{x}) \in Q[\vec{x}]$ are terms of the elementary algebra.

(ii) The comparison predicates $\geq, =$ etc. can be expressed by a combination of logical junctors and $<$.

(iii) The system $\mathcal{U} = \{g_1, \ldots, g_m\}$ has a solution in $\mathbb{R}$ iff $\exists ... \exists\, x_1, \ldots, x_n\ (g_1 \wedge \cdots \wedge g_m)$ is true.

(iv) The negation of an inequality is simply the negation of its comparing operator.

(v) $\exists x\ (x = x)$ is an example for $\top$.

Theorem 3.4
Systems of (in)equalities between multivariate polynomials of arbitrary degree are an admissible concrete domain.
Systems of (in)equalities between linear multivariate polynomials are an admissible concrete domain.

4 State of the art

What is the state of the art in implemented DL systems compared to these results? To answer this we analyse the expressiveness of the abstract as well as the concrete domain of Loom [10], Classic [12], Kris [2] and Taxon [3][7]. It turns out, that at best, the systems are only able to handle numerical constants.
Loom and Classic use structural subsumption algorithms. These algorithms are efficient and sound, but incomplete for expressive description languages. In contrast to that, Kris and Taxon provide sound and complete algorithms based on a model-generating procedure [14] for testing the consistency of an ABox [6, 1]. All systems provide different description languages and inferences, preventing a comparison of the systems as well as the transfer of descriptions from one system to another. Within the Krss project, a concrete and an abstract syntax for a very expressive language description logic was developed [13]. Additionally it defines a core of this language on which compliant implementations are required to implement complete algorithms. We use Krss as a reference for our comparisons.

4.1 Expressiveness of the Abstract Domains

Table 1 gives an overall impression of the expressiveness of the concept terms in the abstract domain of various systems. Since the Krss specification was mainly done by the developers of Classic, it is not surprising that the Krss core and Classic are nearly identical. Kris adds number-restrictions to the language of Taxon . By far the most

[7] There a quite a few other DL systems. We have chosen this selection, cause its a representative cross-section. Furthermore they are freely available (along with their source code) and written in Lisp.

KRSS		CORE	LOOM	CLASSIC	KRIS	TAXON
$\top$	top	$\oplus$	$\odot$	$\odot$	$\odot$	$\odot$
$\bot$	bottom	$\oplus$	$\odot$	$\odot$	$\odot$	$\odot$
$C \sqcap D$	and	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
$C \sqcup D$	or		$\oplus$		$\oplus$	$\oplus$
$\neg C$	not		$\oplus^b$		$\oplus$	$\oplus$
$\forall R : C$	all	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
$\exists R$	some	$\oplus$	$\oplus$		$\oplus$	$\odot$
$\uparrow R$	none		$\odot$		$\odot$	$\odot$
$\geq_n R$	at-least	$\oplus^a$	$\oplus$	$\oplus$	$\oplus$	
$\leq_n R$	at-most	$\oplus^a$	$\oplus$	$\oplus$	$\oplus$	
$=_n R$	exactly	$\oplus^a$	$\odot$	$\odot$	$\odot$	
$\exists R : C$	some		$\oplus$		$\oplus$	$\oplus$
$\geq_n R : C$	at-least		$\odot$			
$\leq_n R : C$	at-most		$\odot$			
$=_n R : C$	exactly		$\odot$			
$R_1 = R_2$	equal	$\oplus^c$	$\oplus$	$\odot^c$	$\oplus^c$	$\oplus^c$
$R_1 \neq R_2$	not-equal		$\odot$		$\oplus^c$	$\oplus^c$

$\oplus$ explicitly contained $\odot$ implicitly expressible
[a] number restriction extension [b] since 2.1 [c] only attributes

Table 1: Expressiveness: Concept Terms

expressive language is provided by LOOM . But its algorithms are incomplete even where complete ones can be given, e.g. with respect to or and not.

4.2 Expressiveness of the concrete domains

KRSS provides a *concrete part* of the domain, specified as the rationals and strings over some alphabet of size at-least 2. Since our example only needs the numeric domain and this is what most of the systems provide, we restrict our analysis to it.

4.2.1 Numeric Base Type

Regarding the numeric base type, number is only a non-terminal in the syntax definition of CLASSIC , but looking at the manual one can find references to reals. LOOM accepts reals in numerical relations while providing the predefined concepts number (including complex numbers) and integer. KRIS is restricted to naturals while TAXON provides rationals.

4.2.2 Syntax

One requirement of Baader and Hanschkes admissible domains is the strict separation between the abstract and the concrete domain, i.e. $dom(\mathcal{I}) \cap dom(\mathcal{D}) = \emptyset$. Therefore KRIS and TAXON restrict the use of expressions of the concrete domain to quantifications

System	Realisation
Krss	(some radius (and (minimum 0) (maximum 20)))
Loom	(:and (>= radius 0) (<= radius 20))
Classic	(all radius (and (min 0) (max 20)))
Kris	(and (>= radius 0) (<= radius 20))
Taxon	(cpred 0-20 (ro (?x) (and (>= ?x 0) (<= ?x 20)))) (some (radius) 0-20)

Table 2: The use of concrete domains in the various systems

over attribute chains. This makes it impossible to define a concept describing the interval [0 20] as it is allowed by Krss , Loom , and Classic . Taxon carries the separation between abstract and concrete domain along to the syntactical level. One first has to define the predicates of the concrete domain with cpred before they can be used in concept terms. Table 2 shows how the range of the attribute radius could be restricted in the various systems.

4.2.3 Expressiveness

Table 3 summarises the systems' capabilities to handle systems of (in)equations, using the different constructs of the elementary algebra over the reals as a guideline. The first and foremost observation is the *complete lack of any function symbols*. Therefore, current terminological systems can at best handle arbitrary comparisons between attributes or attributes and numbers. Only Loom defines a relation +, but it cannot be used within concept definitions. Therefore it is impossible to define even the simplest equations between univariate linear polynomials like $x = y + 1$ or $diameter = 2 \cdot radius$. Not to mention linear multivariate polynomials like $perimeter = 2 \cdot (length + width)$ or nonlinear polynomials like $torque = radius \cdot force$ as in Equation (3). Krss and Classic show another notable limitation. Unlike the other systems they provide only $\geq$ and $\leq$ as comparison operators, and one term of the equation is restricted to be a constant. This makes it impossible to define a square as a special rectangle with $length = width$. One has to use the abstract equality predicate equal in order to describe this. No system implements logical operators or quantifiers within the concrete domain. They rely on the operators of the abstract domain. Because Krss and Classic only provide and, it is impossible to define the missing comparison operator $<$.

The only way to circumvent the above limitations in some way is the following: Krss , Loom , and Classic provide constructs (satisfies, test-h/c) to include arbitrary Lisp functions into concept descriptions. Obviously this construct cannot be used at all within the TBox reasoning services, since that would require the decidability of the halting problem. Therefore, in our opinion these constructs are nearly unusable, especially when it is possible to implement sound and complete algorithms as for systems of inequations of polynomials. W.r.t. ABox reasoning they also have the major disadvantage of being lisp functions and not constraints. Therefore they can act only as test predicates. They

	Core	Loom	Classic	Kris	Taxon
Basetype	$\mathbb{Q}$,$\mathbb{Z}$	$\mathbb{R}$,$\mathbb{Z}$	$\mathbb{R}$	$\mathbb{N}$	$\mathbb{Q}$
Terms					
Constants a	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
Variables x	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
$(x+a)$					
$(x+y)$					
$(a \cdot x)$					
$(x \cdot y)$					
Formulas					
$(\alpha < \beta)$		$\oplus$		$\oplus$	$\oplus$
$(\alpha > \beta)$		$\oplus$		$\oplus$	$\oplus$
$(\alpha = \beta)$	$\odot$	$\oplus$	$\odot$	$\oplus$	$\oplus$
$(\alpha \neq \beta)$		$\oplus$		$\oplus$	$\oplus$
$(\alpha \leq \beta)$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
$(\alpha \geq \beta)$	$\oplus$	$\oplus$	$\oplus$	$\oplus$	$\oplus$
Logical Operators					
$\neg\varphi$		$\otimes$		$\otimes$	$\otimes$
$(\varphi \vee \psi)$		$\otimes$		$\otimes$	$\otimes$
$(\varphi \wedge \psi)$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
Quantifiers					
$(\forall x \varphi)$	$\otimes$	$\otimes$	$\otimes$	$\otimes$	$\otimes$
$(\exists x \varphi)$		$\otimes$			$\otimes$

$\oplus$ explicitly contained $\odot$ implicitly expressible
$\otimes$ via the abstract domain

Table 3: Expressiveness: Linear Sentences

do not propagate restrictions like constraints nor do they expose the undirectedness of relations[8].

5 Ctl

The analysis in the previous section shows the shortcomings of current DL systems w.r.t. systems of (in)equalities. In this section we first give a principled way how to extend a terminological logic with a concrete domain. We then show how CLP systems provide the decision procedure for a whole range of concrete domains and enable us to handle systems of (in)equalities.
Realising a concrete domain is mainly the task of integrating a decision procedure for finite conjunctions of predicates (see Definition 3.2 and Theorem 3.4). In general there are two ways to do so:

1. Realising the decision procedure through a module *hardwired* to the terminological inference engine.

[8] When not explicitily programmed that way.

2. Combining an existing *implementation* of the decision procedure through a *well defined interface* to the terminological inference engine[9].

With solution 1, every modification of the concrete domains, e.g. addition of a new domain, extending the expressiveness or exchanging the decision procedure, leads to a modification of the DL inference engine. With solution 2 this is reduced to the adaptation of the decision procedure to the interface. To our knowledge, all existing terminological systems (e.g. the systems of the previous section) use approach 1. CTL is unique by using approach 2 to provide a generic interface for concrete domains. The interface allows us to simultaneously provide multiple concrete domains, resulting in a modular and configurable terminological system. Due to this open architecture, CTL can be adapted to the needs of the application, by only instantiating the concrete domains that are actually needed.

5.1 The Interface

The abstract domain of CTL is based on $\mathcal{ALC}$ and comparable to TAXON [10]. The inference engine uses a model-generating consistency test [7], resulting in sound and complete algorithms. We use the syntax of the KRSS specification [13]. Definition 3.4 serves as a guideline for the design of the interface. The interface itself is two-layered: First, means for defining concrete predicates and distinguishing between different concrete domains must be introduced at the representational level. Second, a set of functions realising the interface at the inference level must be defined.

5.1.1 The representational level

The concrete part of KRSS is fixed (see Section 4 and [13]). Since we want to be flexible w.r.t. concrete domains we have to find a way to do so. Instead of defining new constructs for each new predicate (like KRSS does with **minimum** or **maximum**[11]), we provide a formalism for introducing new predicate descriptions, similar to the description of concepts or roles.

SYNTAX	SEMANTIC
($\langle \mathsf{name}_D \rangle$ $(X_1 \ldots X_n)$ $\langle \mathsf{expr}_D \rangle$)	predicate term (see text)
(define-predicate PN P)	$PN^{\mathcal{I}} = P^{\mathcal{I}}$
(constrain $\mathsf{R}_1 \ldots \mathsf{R}_n$ P)	$\{a \mid \exists b_1, \ldots, b_n : (a, b_1) \in R_1^{\mathcal{I}} \wedge \cdots \wedge (a, b_n) \in R_n^{\mathcal{I}} \wedge (b_1, \ldots, b_n) \in P^{\mathcal{I}}\}$

Table 4: Extensions: Representational level

Predicate terms P describe concrete predicates. They are either a predicate name PN or a list ($\langle \mathsf{name}_{\mathcal{D}} \rangle$ $(x_1 \ldots x_n)$ $\langle \mathsf{expr}_{\mathcal{D}} \rangle$) consisting of a domain identifier $\langle \mathsf{name}_{\mathcal{D}} \rangle$ and a list

[9] This follows the basic ideas presented by Baader and Hanschke in [1].

[10] adding the missing number-restrictions is simply the task of adding rules to the inference engine

[11] minimum and maximum are provided for backward compatibility and handled as predefined concrete predicates.

of variables $x_1 \dots x_n$. The expression $\langle \mathsf{expr}_{\mathcal{D}} \rangle$ is written in a syntax understood by the decision procedure of the concrete domain $\mathcal{D}$, actually defines the concrete predicate and refers to the variables. define-predicate assigns a name PN to a predicate term P. Because concrete predicates constrain the fillers of relations (i.e. roles and attributes), predicate terms are allowed in concept as well as relation descriptions. We introduce constrain as an additional concept term with the following semantic: For every combination $(d_1, ..., d_n)$ of fillers of the relations $R_1, ..., R_n$ the predicate term P will be instantiated and the fillers accordingly restricted. Within role and attribute terms the R_i are restricted to chains beginning with domain or range.

5.1.2 The inference level

On the interface level, the terminological inference engine has to be extended in order to communicate with the decision procedure of the concrete domain. constrain constructs are handled by the concrete domain rules of the model-generating procedure [1, 5]. Each time such a rule is called, the inference engine has to determine all combinations of the fillers and to instantiate the expression of the predicate term by substituting every variable x_i with the respective filler d_i. Second, a pool of the instantiated expressions must be maintained. Each time an expression is instantiated, it is added to the pool (add-expr) and the decision procedure of the concrete domain is called through consistent, to determine whether the extended pool is consistent. In the case of an inconsistency, expressions may be removed from the pool (rem-expr) if there are any choice points to backtrack to. Third, during the inferences the inference engine may need the negation of a concrete predicate. Therefore, the concrete domain must provide a function negation for negating concrete predicates (e.g. their expressions).
To summarise, a concrete domain interface has to provide a decision procedure, functions for administrating the expression pool (minimal functions to add and remove expressions) and function to convert expressions to their negated form.

5.2 CLP-systems as decision procedures

Our example from section 2 needs a concrete domain handling systems of (non)linear (in)equalities. The basic idea is to realise this domain by interfacing to CLP($\mathcal{R}$)-systems. In our current implementation we interfaced to a CLP($\mathcal{R}$)implementation called EPILYTIS (derived from [9]) by extending it with functions for negation and pool management.
This instantly gives us sound and complete inferences over arbitrary linear polynomials, considerably improving the expressiveness of the resulting system. In section 2 we have seen that the handling of Equation 3 within the TBox requires nonlinear constraints. Nonlinear constraints are handled within CLP($\mathbb{R}$) systems if they become linear during the inference process.
Overcoming this limitation is one of the central themes in the areas of CLP as well as computer algebra. Systems like GDCC [16] make use of the Buchberger algorithm for computing Gröbner bases in order to solve the problem over the complex numbers instead of the reals. While still being incomplete this significantly enhances the expressiveness of

the system. Risc-CLP(Real) [8] is the first CLP-system delivering complete inferences for arbitrary systems of inequalities, by improving Collins technique for finding cylindric algebraic decompositions[12].
With Ctl we automatically take advantage of these and any future improvements, while this would at least be arduous if not impossible in other systems. Furthermore, interfacing to CLP-systems can be used to realise a whole range of concrete domains, e.g. finite domains (i.e. sets of symbols) with CLP(FD)[13].

6 Realising the example with Ctl

In order to realise the example we first have to define an appropriate TBox[14]. This is done in three steps:

1. First we have to define the concrete predicates underlying the physical laws (1 - 5),

```
(define-predicate X=Y (EPILYTIS (?X ?Y) (?X = ?Y)))
(define-predicate X*Y=Z (EPILYTIS (?X ?Y ?Z)
  (?X * ?Y = ?Z)))
(define-predicate X>=0 (EPILYTIS (?X) (?X >= 0)))
(define-predicate X>0 (EPILYTIS (?X) (?X > 0)))
```

2. This is followed by the definition of concepts describing the components: general, rotational and tension links.

```
(define-primitive-attribute link.force TOP)
(define-primitive-attribute rot.radius TOP)
(define-primitive-attribute rot.torque TOP))
(define-primitive-concept link
  (some link.force (minimum 0)))
(define-primitive-concept r-link
  (and link
    (some rot.radius X>0)
    (some rot.torque X>=0)
    (constrain rot.radius link.force  rot.torque
      X*Y=Z)))
(define-primitive-concept t-link link)
```

3. Third, we have to describe connections between components. We represent them as roles. Note that the physical laws (1) and (2) describe effects taking place on certain kinds of connections.

[12] This technique can be applied to the problem of quantifier elimination on which Tarskis decision procedure is based.
[13] We already have done that, but this is beyond the scope of this paper.
[14] Due to space constraints, we provide a minimal TBox.

```
(define-role linked
  (and (domain link) (range link)))
(define-role linked.rot.rot
  (and (domain r-link)
       (range r-link)
       (constrain (compose range rot.torque)
                  (compose domain rot.torque) X=Y)))
(define-role linked.rot.ten
  (and (domain r-link) (range t-link)
       (constrain (compose range link.force)
                  (compose domain link.force) X=Y)))
(define-role linked.ten.rot
  (and (domain t-link) (range r-link)
       (constrain (compose range link.force)
                  (compose domain link.force) X=Y)))
```

The Abox describing the drivetrain is then instantiated in 4 steps:

1. First we have to define the individuals[15]. Note, that the instantiation of the individuals already leads to the instantiation and restriction of their attributes.

```
(state (and
  (instance ca-1 r-link) (instance cr-1 r-link)
  (instance bbs-1 r-link) (instance ch-1 t-link)
  (instance sp-1 r-link) (instance ra-1 r-link)
  (instance rw-1 r-link)))
```

2. Then we establish the connections between the individuals.

```
(state (and
  (related ca-1  bbs-1 linked.rot.rot)
  (related bbs-1 cr-1  linked.rot.rot)
  (related cr-1  ch-1  linked.rot.ten)
  (related sp-1  ch-1  linked.rot.ten)
  (related sp-1  ra-1  linked.rot.rot)
  (related rw-1  ra-1  linked.rot.rot)))
```

If we now define the values of some attributes, CTL propagates these values through the interface to the CLP($\mathcal{R}$)–system. The CLP($\mathcal{R}$)-system checks the consistency of the (in)equality system and calculates the values for all other attributes:

3. First we give the radii of the rotational links.

[15]The names of the individuals are abbreviated as in Figure 1.

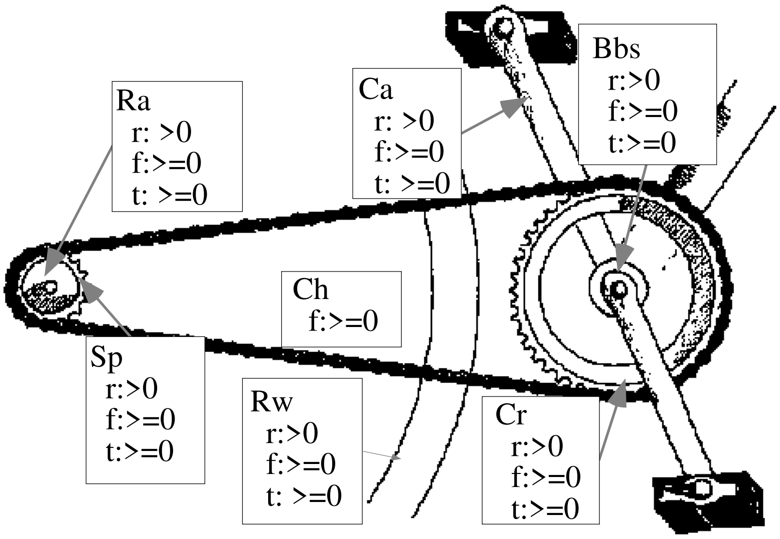

Figure 2: The generated model after step 1.

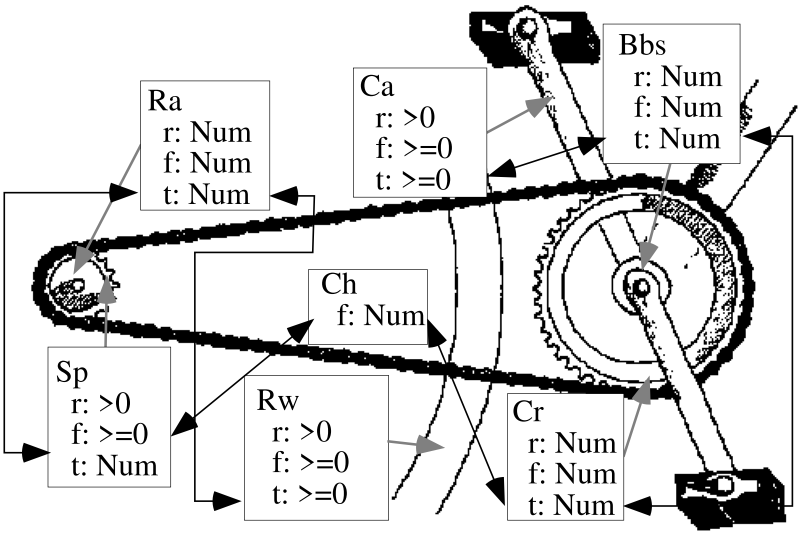

Figure 3: The generated model after step 2.

```
(state (and (related ca-1 .175   rot.radius)
            (related cr-1 .1     rot.radius)
            (related sp-1 .05    rot.radius)
            (related rw-1 0.6858 rot.radius)))
```

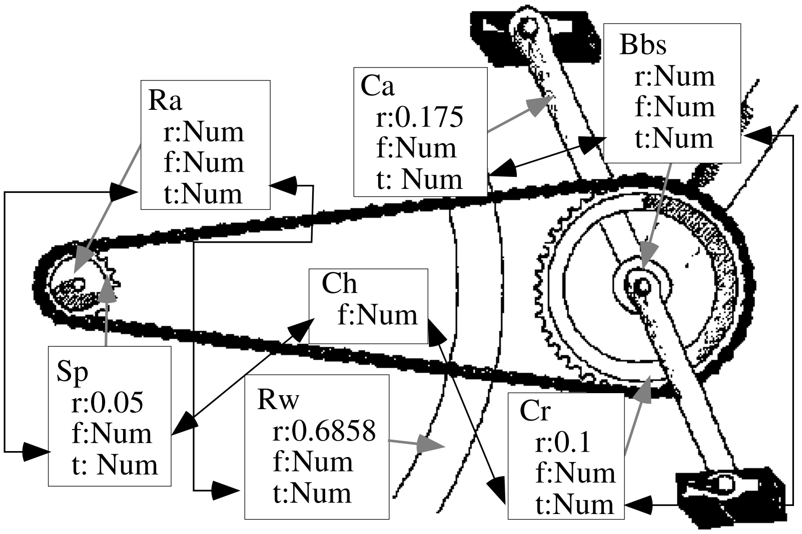

Figure 4: The generated model after step 3.

4. Finally we assert the value of the force applied to the crank arm.

   ```
   (state (related ca-1 200    link.force))
   ```

   This leads to the propagation of the constraints and the values of the other attributes are determined[16].

7 Summary and outlook

The example of a bike drivetrain reveals that in order to represent physical laws, terminological systems need to be able to handle systems of (non)linear (in)equalities. This is possible without losing the ability to define sound and complete algorithms. We analysed a representative set of current DL systems and found that they are way too inexpressive. The shortcomings of existing terminological systems lead us to the development of a new system called CTL . CTL is based on a terminological system with a generic interface for concrete domains [1], allowing the easy attachment of existing decision procedures. Examples for such decision procedures are existing CLP-systems. In particular we attached a CLP($\mathcal{R}$)–system for the bike drivetrain example.

[16]The values are typical ones for a racing bike and an average cyclist.

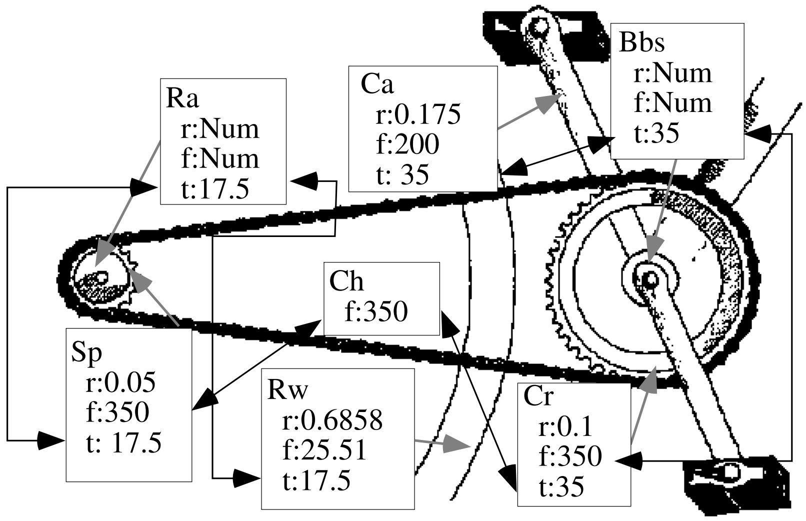

Figure 5: The generated model after step 4.

Future work is concentrated on attaching more CLP-systems. Currently we are working on the integration of R. We plan to attach non-linear CLP($\mathcal{R}$)-systems (e.g. [8]) as soon as they are available.

References

[1] Baader, F., and Hanschke, P. A scheme for integrating concrete domains into concept languages. Tech. Rep. RR-91-10, DFKI, Apr. 1991.

[2] Baader, F., and Hollunder, B. Kris: Knowledge representation and inference system – system description. Technical Memo TM-90-03, DFKI, 11 1990.

[3] Boley, H., Hanschke, P., Hinkelmann, K., and Meyer, M. CoLab: A Hybrid Knowledge Representation and Compilation Laboratory. Tech. Rep. RR-93-08, DFKI, Jan. 1993.

[4] Frühwirth, T., and Hanschke, P. Terminological reasoning with constraint handling rules. In *Principles & Practice of Constraint Programming*. MIT Press, Cambdrige, MA, 1995, ch. 19, pp. 361–381.

[5] Hanschke, P. *A Declarative Integration of Terminological, Constraint-Based, Data-driven, and Goal-directed Reasoning*. Dissertation, Universität Kaiserslautern, 1993.

[6] Hollunder, B. Hybrid inferences in kl-one-based knowledge representation systems. In *Proc. GWAI-90* (Ehringerfeld, Germany, 1990), Informatik Fachberichte, Springer.

[7] Hollunder, B., and Nutt, W. Subsumption algorithms for concept languages. Tech. Rep. RR-90-04, DFKI, Apr. 1990.

[8] Hong, H. RISC-CLP(Real): Constraint Logic Programming over the real numbers. In *Constraint Logic Programming: Selected Research*. MIT Press, Cambridge, MA, 1993.

[9] Jaffar, J., Michaylov, S., Stuckey, P. J., and Yap, R. H. The CLP(R) Language and System,.

[10] MacGregor, R. The evolving technology of classification-based knowledge representation systems. In *Principles of Semantic Networks*. Morgan Kaufmann, 1991.

[11] MacGregor, R. M. A description classifier for the predicate calculus. In *Proc. of the AAAI-94* (1994), AAAI, pp. 213–220.

[12] Patel-Schneider, P., McGuiness, D., Brachman, R., Resnick, L., and Borgida, A. The CLASSIC Knowledge Representation System: Guiding Principles and Implementation Rational. *SIGART Bulletin 2*, 3.

[13] Patel-Schneider, P. F., and Swartout, B. Description logic specification from the krss effort. Nov. 1993.

[14] Schmidt-Schauss, M., and Smolka, G. Attributive concept descriptions with complements. *AI 47* (1991).

[15] Tarski, A. A decision method for elementary algebra and geometry. In *Collected Works of A. Tarski*, vol. 3. pp. 297–368,.

[16] Terasaki, S., Hawley, D. J., Sawada, H., Satoh, K., Menju, S., Kawagishi, T., Iwayama, N., and Aiba, A. Parallel constraint logic programming language GDCC and its parallel constraint solvers. In *International Conference on Fifth Generation Computer Systems* (June 1992).

CTL
A description logic with expressive concrete domains

Gerd Kamp and Holger Wache

LKI-M-96/01
Technical Report